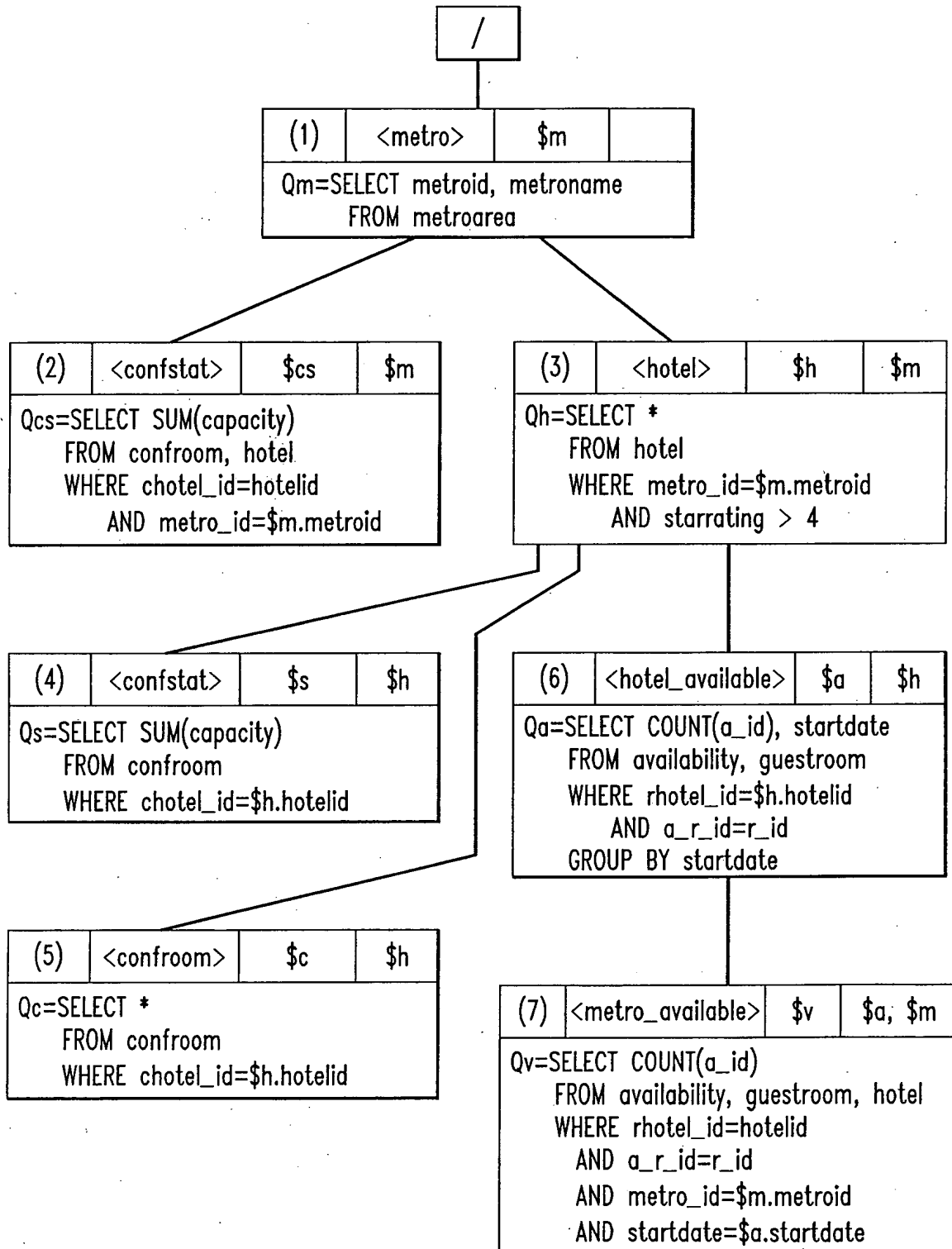




1/22

FIG. 1

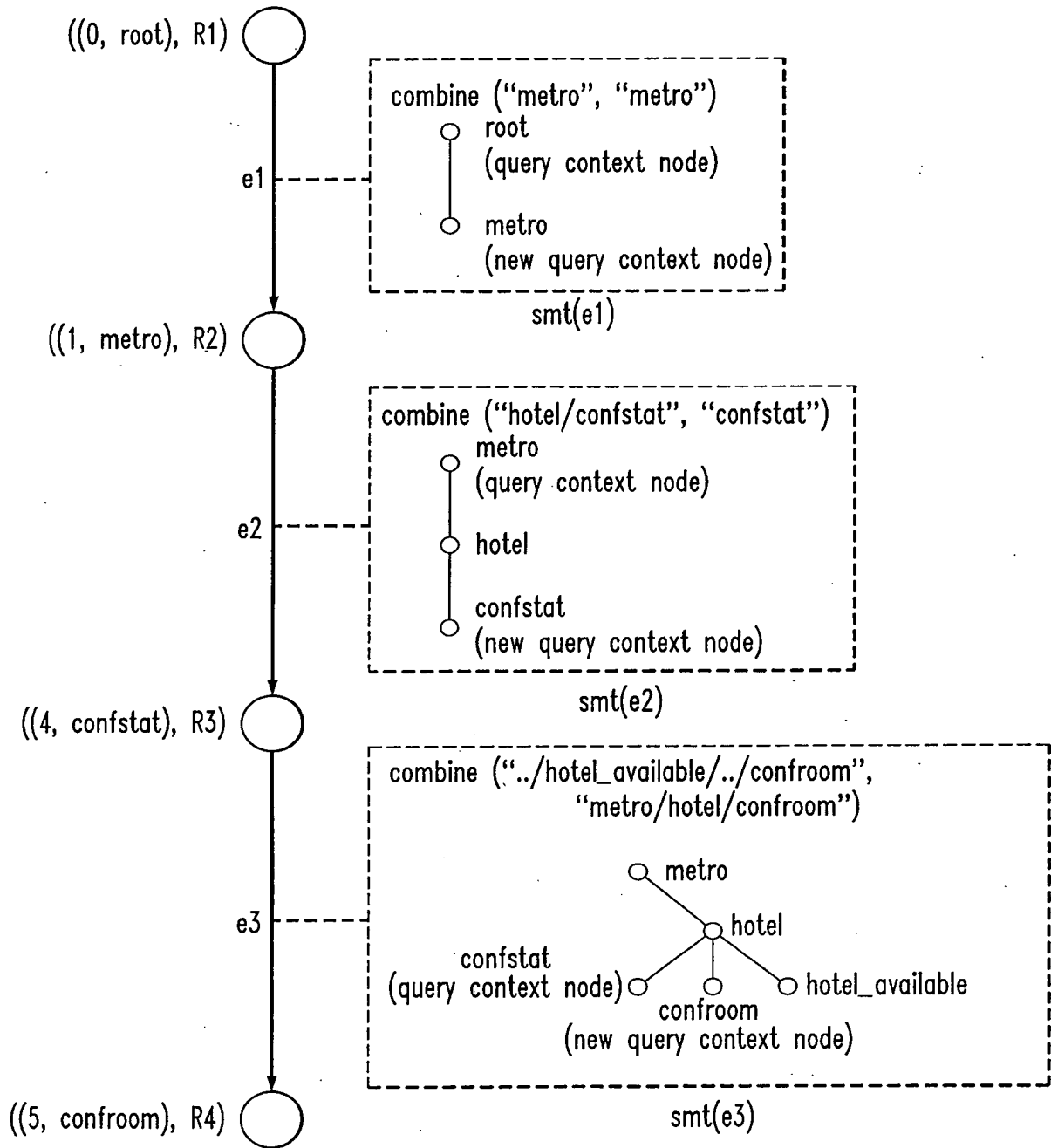
100



5/22

FIG. 8

800
↙

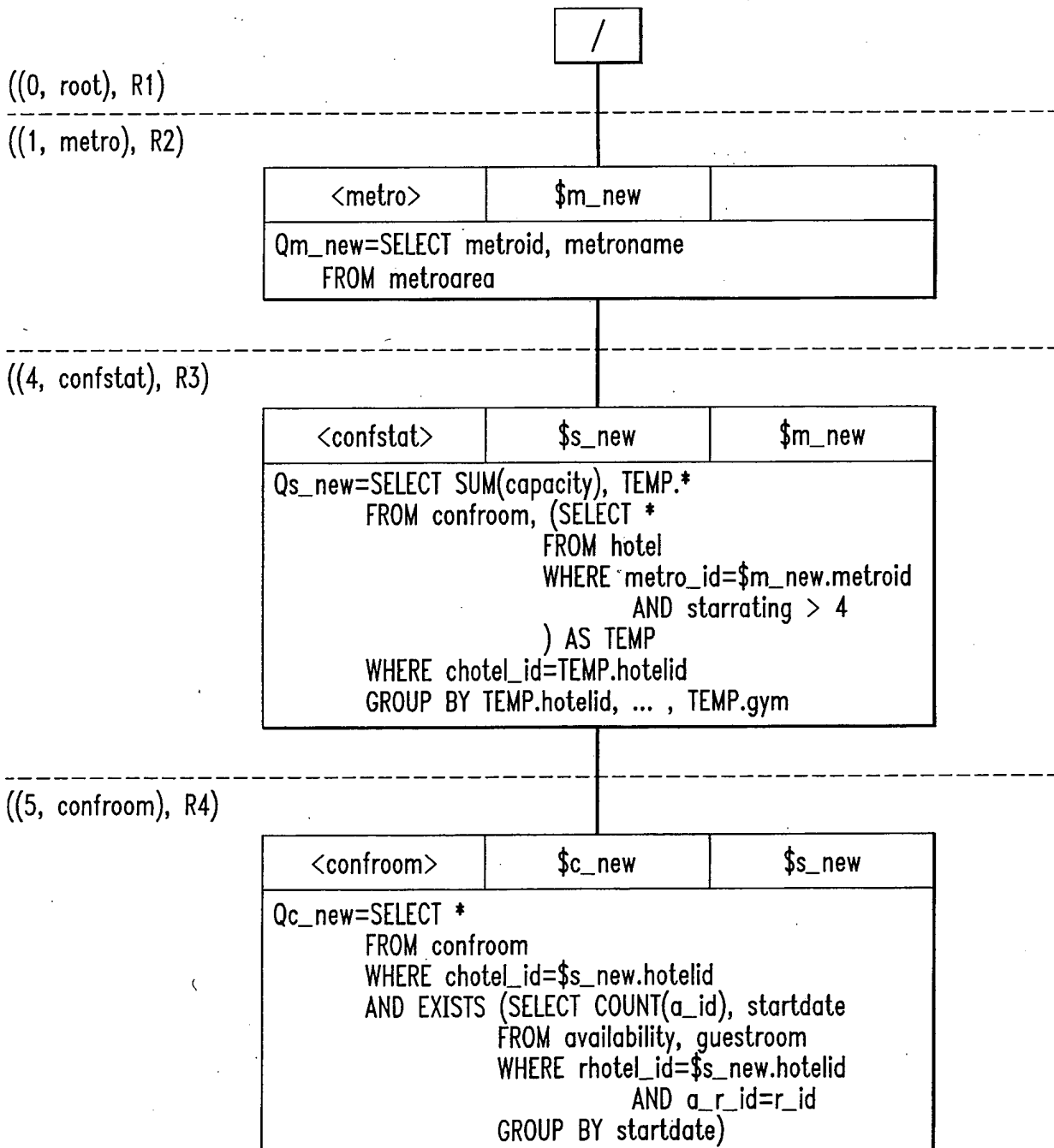


6/22

FIG. 9

(a) TRAVERSE VIEW QUERY

900
 ↓

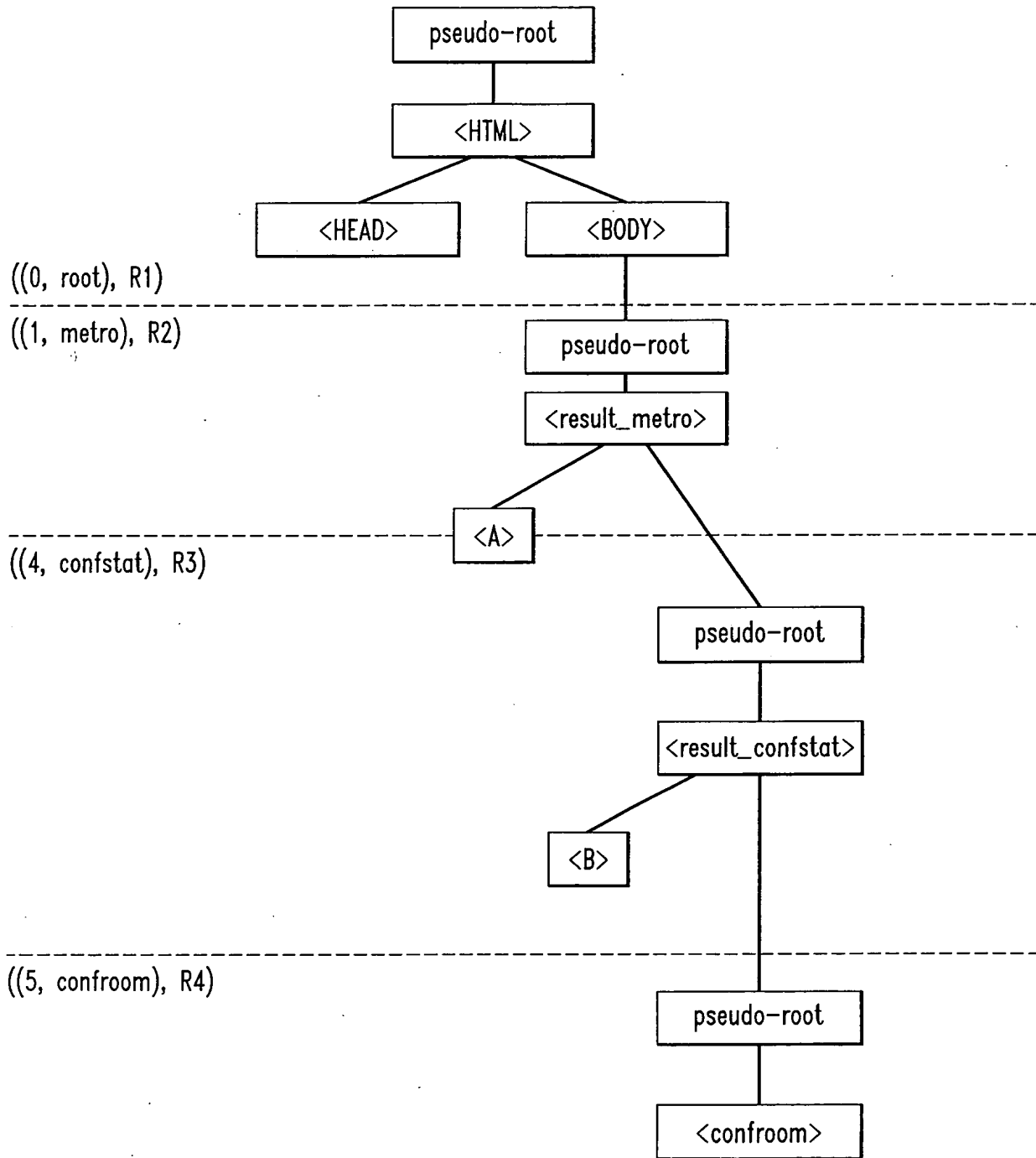


7/22

FIG. 10

(b) OUTPUT TAG TREE

1000
2

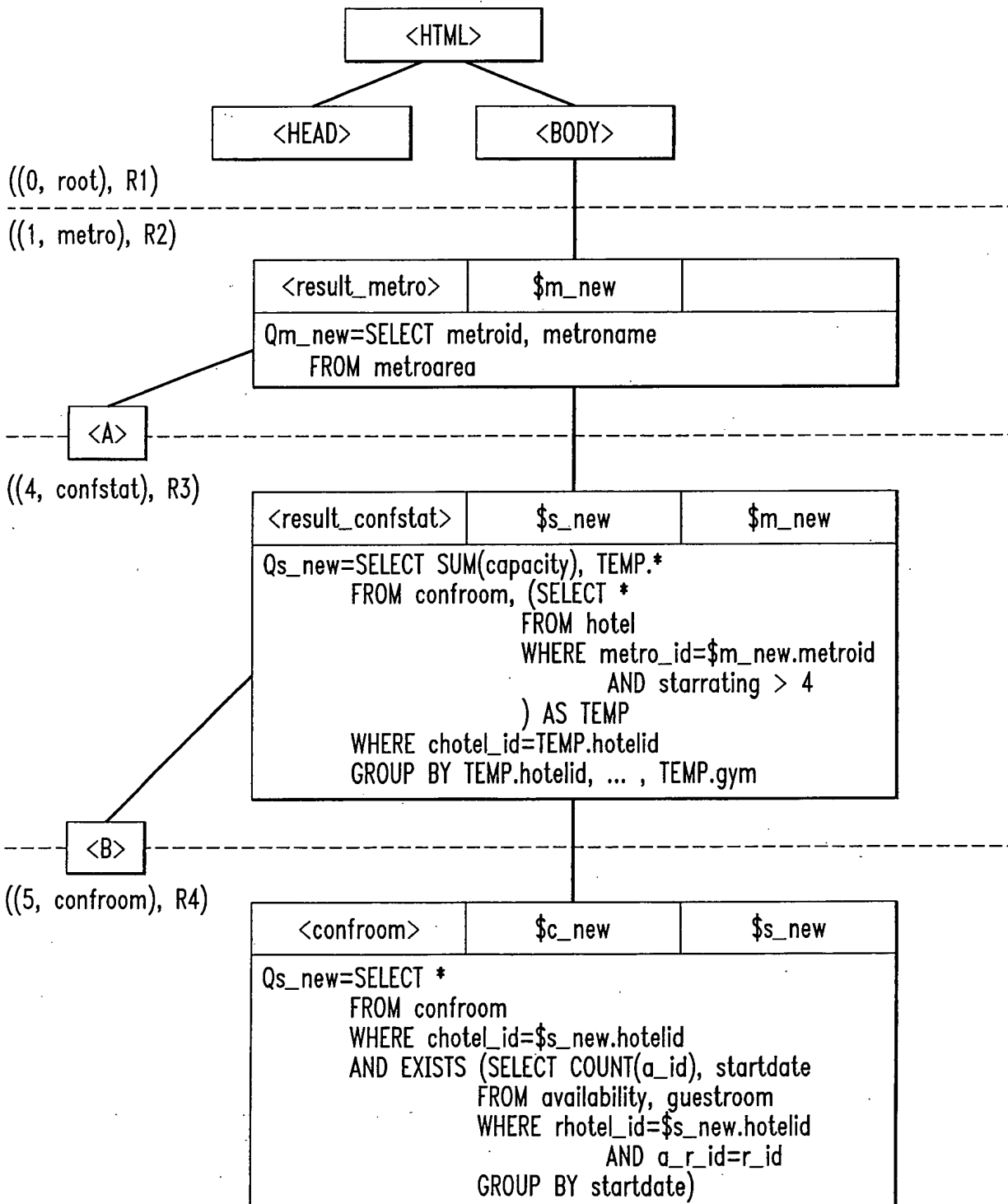


8/22

FIG. 11

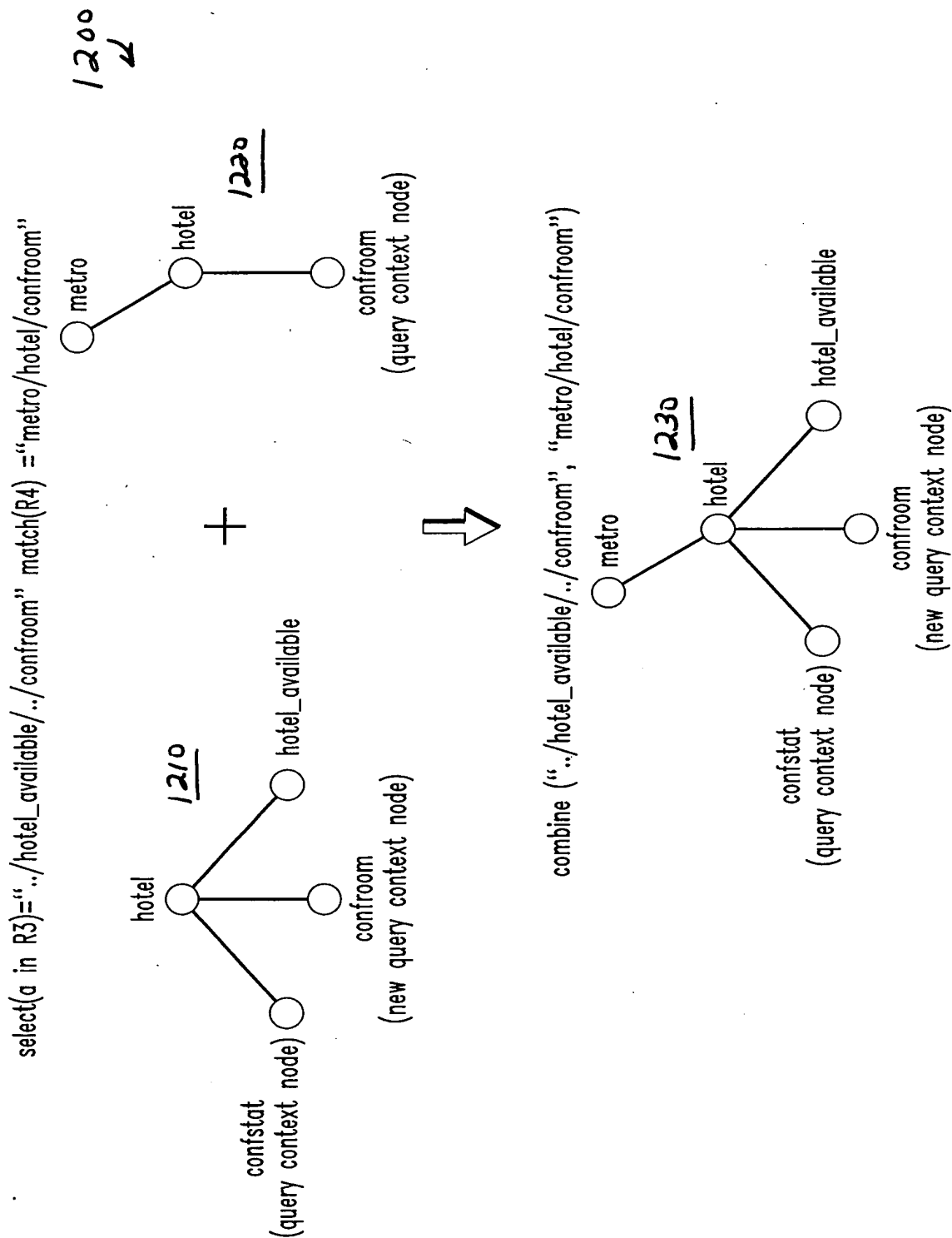
(c) STYLESHEET VIEW

1100

9/22

FIG. 12



10/22

FIG. 13

~ 1300

Procedure *Compose*(v, x)

Input: v : original schema-tree view query; x : XSLT stylesheet

Output: stylesheet view

1: ctg : a Context Transition Graph

2: tvq : a Traverse View Query

3: $ottree$: an Output Template Tree

4: **for** $n \in v$ **do**

5: **for** $r \in x$ **do**

6: **if** $MATCHQ(n, r) \neq \text{NULL}$ **then**

7: add (n, r) to ctg

8: **for** $(n_1, r_1) \in ctg$ **do**

9: **for** $(n_2, r_2) \in ctg$ **do**

10: **for** $a \in \text{apply}(r_1)$ **do**

11: $t \leftarrow SELECTQ(n_1, a, n_2)$, $p \leftarrow MATCHQ(n_2, r_2)$

12: **if** $t \neq \text{NULL}$ **and** $\text{mode}(a) = \text{mode}(r_2)$ **then**

13: add an edge $e = ((n_1, r_1), (n_2, r_2), a)$ to ctg

14: $\text{smt}(e) \leftarrow COMBINE(t, p)$

15: (repeatedly) Delete all nodes without incoming edge, except $(root, r)$

16: $tvq \leftarrow ctg \{(\text{copy})\}$, $\text{bmap}(\text{root of } tvq) \leftarrow \text{empty}$

17: (repeatedly) Duplicate nodes with multiple incoming edges, splitting incoming edges and copying outgoing edges

18: replace binding variables in tvq with new, unique binding variables

19: **for** $e = (w_1 = (n_1, r_1), w_2 = (n_2, r_2), a)$ in edges of ctg **do**

20: $(Q_{bv(w_2)}, \text{bmap}(w_2)) \leftarrow$

$UNBIND(\text{smt}(e), n_1, n_2, bv(w_2), \text{bmap}(w_1))$

21: **for** all binding variables bv referenced in $Q_{bv(w_2)}$ **do**

22: rename bv as $\text{bmap}(w_2).get(bv)$

11/22

FIG. 13 cont.

```
23: for  $w = (n, r) \in tvq$  do
24:    $ott(w) = GENERATE\_OTT(n, r)$ 
25:  $ottree \leftarrow ott(w) \forall w \in tvq$  {initially a forest}
26: for  $e = (w_1, w_2, a)$  in edges of  $tvq$  do
27:    $a' \leftarrow$  the "apply-template" node in  $ott(w_1)$  which is a copy of  $a$ 
28:   replace  $a'$  with an edge from  $parent(a')$  to  $root(ott(w_2))$ 
29: for  $w = (n, r) \in tvq$  do
30:    $bv(root(ott(w))) \leftarrow bv(w)$ 
31:    $Q_{bv(root(ott(w)))} \leftarrow Q_{bv(w)}$ 
32: Remove the topmost "pseudo-root" node in  $ottree$ 
33: while there is any "pseudo-root" node  $pr$  left do
34:   for each child node  $c$  of  $pr$  do
35:     add edge  $e = (parent(pr), c)$ 
36:     if  $Q_{bv(c)}$  is empty then
37:        $bv(c) \leftarrow bv(pr)$ ,  $Q_{bv(c)} \leftarrow Q_{bv(pr)}$ 
38:     else
39:        $Q_{bv(c)} \leftarrow UNBIND(parent(pr), c)$ 
40:       add the SELECT columns of  $Q_{bv(pr)}$  to  $Q_{bv(c)}$ 
41:       change " $bv(pr)$ " as " $bv(c)$ " in the tag queries of  $c$ 's
         descendants
42:   remove edge  $e = (parent(pr), pr)$  and node  $pr$ 
43: return  $ottree$  {new stylesheet view}
```


12/22

FIG. 14

~1400

Function *UNBIND*(m, n)
Input: Query context node as m , Target node to unbind as n
Output: Unbound query for n as q
1: $n_j \leftarrow$ the lowest common ancestor of m and n
2: $s_j \leftarrow bv(n_j)$
3: $child_n(n_j) \leftarrow$ child node of n_j along the path from n_j to n
4: $s_{j-1} \leftarrow bv(child_n(n_j))$
5: $q \leftarrow Q_{bv(n)}^{s_1, s_2, \dots, s_{j-1}}(s_j, \dots, s'_k)$, which is the unbound tag query of n , $Q_{bv(n)}(s_1, s_2, \dots, s_k)$
6: **return** q

FIG. 15

Function *NEST*(p, p')
Input: Node as p , Child node as p'
Output: Tag query for p after nesting as q
1: $q \leftarrow Q_{bv(p)}$
2: **for each** child node c of p , except p' **do**
3: $q_c \leftarrow NEST(c, NULL)$
4: add *EXISTS* q_c in *WHERE* clause of q
5: **return** q

FIG. 16

1: {Replace line 5 of Figure 10}
2: $P \leftarrow$ {nodes along the path from $child_n(n_j)$ to n }
3: **for** node $p \in P$ **do**
4: $p' \leftarrow$ child of $p \in P$, otherwise *NULL*
5: $\Theta_{bv(p)} \leftarrow NEST(p, p')$
6: $q \leftarrow \Theta_{bv(n)}^{s_1, s_2, \dots, s_{j-1}}(s_j, \dots, s'_k)$, which is the unbound and nested tag query of n , $\Theta_{bv(n)}(s_1, s_2, \dots, s_k)$, and decorrelation of $bv(s_i)$ is done by Θ_{s_i} .